ASSIGNMENT 18

# Q1

Text files and binary files differ in how they store and represent data. Text files contain human-readable characters encoded in a specific character encoding, such as ASCII or UTF-8. They are typically used to store plain text documents, configuration files, or structured data in a human-readable format. Binary files, on the other hand, store data in a binary format that is not directly human-readable. They can store complex data structures, images, audio files, executables, and other types of data that require a specific binary representation.

# Q2

Text files are often preferred when the data is meant to be human-readable or editable. They are suitable for storing textual data, such as documents, log files, or configuration files, where the content needs to be easily readable and modifiable by humans. Binary files are typically used when storing non-textual data, such as images, audio, video, or serialised objects, where the exact binary representation is important and human readability is not a requirement.

# Q3

When using binary operations to read and write a Python integer directly to disk, one of the main issues is that the binary format may not be platform-independent. Different platforms may use different byte orderings (big-endian or little-endian), resulting in compatibility issues when reading or writing integers between platforms. Additionally, the size of the integer data type may vary between platforms. These differences can lead to data corruption or incorrect interpretation of the integer values when working with binary files across different platforms.

# Q4

Using the with keyword to open a file has the benefit of automatically handling the file's cleanup and closure. The with statement ensures that the file is properly closed after the block of code finishes executing, even if an exception occurs. This helps prevent resource leaks and simplifies the code by eliminating the need for explicit file.close() calls. The with statement improves code readability and ensures proper file handling without the risk of accidentally leaving files open.

# Q5

When reading a line of text in Python, the trailing newline character (if present) is included in the returned line. However, when writing a line of text using the write() method, Python does not automatically append a newline character. It's up to the programmer to explicitly include the newline character (\n) if desired.

# Q6

The file operations that enable random-access operations in Python are seek() and tell(). The seek() function allows us to change the current position (offset) within the file, enabling random access to specific parts of the file. The tell() function returns the current position (offset) within the file. Together, these operations allow us to read or write data at specific positions in the file, providing random access functionality.

# Q7

The struct package in Python is commonly used when dealing with binary data and low-level data structures. It provides functions to convert between Python values and C-style binary data, allowing us to pack and unpack binary data into a specific format. The struct package is useful in scenarios where we need precise control over the binary representation of data, such as when working with network protocols, file formats, or interacting with low-level systems.

# Q8

Pickling in Python is the process of serialising objects into a binary format that can be stored or transmitted and later deserialized to reconstruct the objects. Pickling is useful when we need to store complex data structures, such as dictionaries, lists, or custom objects, preserving their internal structure and relationships. It allows us to easily save and restore objects without having to manually convert them to a specific file format.

# Q9

The shelve package in Python provides a high-level interface for storing and retrieving Python objects in a persistent dictionary-like format. It combines the capabilities of a dictionary and pickling, allowing us to store and retrieve objects by key. The shelve package is best used when we need a simple way to persistently store and access Python objects, such as caching computed results, managing application state, or storing user preferences.

# Q10

A special restriction when using the shelve package is that the keys used to access objects in the shelf must be strings. The shelve module uses strings as keys to store and retrieve objects, so any non-string keys will be automatically converted to strings before being used. This means that if we intend to use non-string keys, such as integers or tuples, we need to convert them to strings before storing and retrieving them from the shelf.